JASPER REPORT - EXPORTING REPORTS

We have seen in the previous chapter, how to print and view a JasperReport generated document. Here we shall see how to transform or export these reports in other formats like PDF,HTML and XLS. Facade class *net.sf.jasperreports.engine.JasperExportManager* is provided to achieve this functionality. Exporting means transforming the *JasperPrint* object (.jrprint file) into different format.

The following code(JasperReportExport.java) demonstrates the exporting process of the JasperReport document. The JasperExportManager provides methods to export a report into PDF, HTML and XML only. To export to the XLS format we have used the class *net.sf.jasperreports.engine.export.JRXlsExporter*. This code generates following three files:

- sample_report.pdf
- sample_report.html
- sample_report.xls

Exporting to other formats

Let's write a report template. The contents of the JRXML file (C:\tools\jasperreports-5.0.1\test\jasper_report_template.jrxml) are as below:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jasperReport PUBLIC "//JasperReports//DTD Report Design//EN"
"http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports
http://jasperreports.sourceforge.net/xsd/jasperreport.xsd"
name="jasper_report_template" language="groovy" pageWidth="595"
pageHeight="842" columnWidth="555" leftMargin="20" rightMargin="20"
topMargin="20" bottomMargin="20">

    <queryString>
 <![CDATA[]]>
    </queryString>
    <field name="country" >
        <fieldDescription><![CDATA[country]]></fieldDescription>
    </field>
    <field name="name" >
        <fieldDescription><![CDATA[name]]></fieldDescription>
    </field>
    <columnHeader>
     <band height="23">
 <staticText>
     <reportElement mode="Opaque" x="0" y="3" width="535"
  height="15" backcolor="#70A9A9" />
     <box>
         <bottomPen lineWidth="1.0" lineColor="#CCCCCC" />
     </box>
     <textElement />
     <text><![CDATA[]]> </text>
 </staticText>
 <staticText>
     <reportElement x="414" y="3" width="121" height="15" />
     <textElement textAlignment="Center"
         verticalAlignment="Middle">
  <font isBold="true" />
    </textElement>
     <text><![CDATA[Country]]></text>
 </staticText>
```

```
 <staticText>
     <reportElement x="0" y="3" width="136" height="15" />
     <textElement textAlignment="Center"
         verticalAlignment="Middle">
  <font isBold="true" />
     </textElement>
     <text><![CDATA[Name]]></text>
 </staticText>
     </band>
    </columnHeader>
    <detail>
        <band height="16">
  <staticText>
     <reportElement mode="Opaque" x="0" y="0" width="535"
   height="14" backcolor="#E5ECF9" />
     <box>
            <bottomPen lineWidth="0.25" lineColor="#CCCCCC" />
     </box>
     <textElement />
     <text><![CDATA[]]> </text>
  </staticText>
  <textField>
     <reportElement x="414" y="0" width="121" height="15" />
     <textElement textAlignment="Center"
         verticalAlignment="Middle">
   <font size="9" />
     </textElement>
     <textFieldExpression >
         <![CDATA[$F{country}]]>
  </textFieldExpression>
  </textField>
  <textField>
     <reportElement x="0" y="0" width="136" height="15" />
     <textElement textAlignment="Center"
         verticalAlignment="Middle" />
         <textFieldExpression >
   <![CDATA[$F{name}]]>
         </textFieldExpression>
  </textField>
       </band>
    </detail>
</jasperReport>
```

Next, contents of the POJO file **C:\tools\jasperreports-5.0.1\test\src\com\tutorialspoint\DataBean.java** are as below:

```
package com.tutorialspoint;

public class DataBean {
   private String name;
   private String country;

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public String getCountry() {
      return country;
   }

   public void setCountry(String country) {
      this.country = country;
   }
}
```

The contents of file **C:\tools\jasperreports-5.0.1\test\src\com\tutorialspoint\DataBeanList.java** are as below:

```
package com.tutorialspoint;

import java.util.ArrayList;

public class DataBeanList {
   public ArrayList<DataBean> getDataBeanList() {
      ArrayList<DataBean> dataBeanList = new ArrayList<DataBean>();

      dataBeanList.add(produce("Manisha", "India"));
      dataBeanList.add(produce("Dennis Ritchie", "USA"));
      dataBeanList.add(produce("V.Anand", "India"));
      dataBeanList.add(produce("Shrinath", "California"));

      return dataBeanList;
   }

   /**
    * This method returns a DataBean object,
    * with name and country set in it.
    */
   private DataBean produce(String name, String country) {
      DataBean dataBean = new DataBean();
      dataBean.setName(name);
      dataBean.setCountry(country);
      return dataBean;
   }
}
```

Write a main class file **JasperReportFill.java**, which gets the java bean collection from the class (DataBeanList) and passes it to the Jasper report engine, to fill the report template. Save it to directory **C:\tools\jasperreports-5.0.1\test\src\com\tutorialspoint**.

```
package com.tutorialspoint;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JRExporterParameter;
import net.sf.jasperreports.engine.JasperExportManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;
import net.sf.jasperreports.engine.export.JRXlsExporter;

public class JasperReportFill {
   @SuppressWarnings("unchecked")
   public static void main(String[] args) {
      String sourceFileName = "c://tools/jasperreports-5.0.1/"
            + "test/jasper_report_template.jasper";
      String printFileName = null;
      DataBeanList DataBeanList = new DataBeanList();
      ArrayList dataList = DataBeanList.getDataBeanList();
      JRBeanCollectionDataSource beanColDataSource =
         new JRBeanCollectionDataSource(dataList);

      Map parameters = new HashMap();
      try {
         printFileName = JasperFillManager.fillReportToFile(sourceFileName,
            parameters, beanColDataSource);
         if (printFileName != null) {
            /**
             * 1- export to PDF
             */
            JasperExportManager.exportReportToPdfFile(printFileName,
                  "C://sample_report.pdf");

            /**
             * 2- export to HTML
             */
```

```
            JasperExportManager.exportReportToHtmlFile(printFileName,
                  "C://sample_report.html");

            /**
             * 3- export to Excel sheet
             */
            JRXlsExporter exporter = new JRXlsExporter();

            exporter.setParameter(JRExporterParameter.INPUT_FILE_NAME,
                  printFileName);
            exporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME,
                  "C://sample_report.xls");

            exporter.exportReport();
         }
      } catch (JRException e) {
         e.printStackTrace();
      }
   }
}
```

Here we have included the logic to export the jasper print file to pdf, html and xls format.

Generating Reports

Let's compile and execute above files using our regular ANT build process. The build.xml file is as below:

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="JasperReportTest" default="executereport" basedir=".">
   <import file="baseBuild.xml"/>

   <target name="executereport" depends="compile,compilereportdesing,run">
        <echo message="Im here"/>
   </target>
    <target name="compilereportdesing"
         description="Compiles the JXML file and
         produces the .jasper file.">
         <taskdef name="jrc"
         classname="net.sf.jasperreports.ant.JRAntCompileTask">
            <classpath ref />
         </taskdef>
         <jrc destdir=".">
            <src>
            <fileset dir=".">
               <include name="*.jrxml" />
            </fileset>
            </src>
            <classpath ref />
         </jrc>
   </target>
</project>
```

Go to the command prompt and then go to the directory C:\tools\jasperreports-5.0.1\test, where build.xml is placed. Finally execute the command **ant -Dmain-class=com.tutorialspoint.JasperReportFill**. The output is as follows:

```
C:\tools\jasperreports-5.0.1\test>ant -Dmain-class=com.tutorialspoint.JasperReportFill
Buildfile: C:\tools\jasperreports-5.0.1\test\build.xml

clean-sample:
   [delete] Deleting directory C:\tools\jasperreports-5.0.1\test\classes
   [delete] Deleting: C:\tools\jasperreports-5.0.1\test\jasper_report_template.jasper
   [delete] Deleting: C:\tools\jasperreports-5.0.1\test\jasper_report_template.jrprint

compile:
    [mkdir] Created dir: C:\tools\jasperreports-5.0.1\test\classes
    [javac] C:\tools\jasperreports-5.0.1\test\baseBuild.xml:28:
    warning: 'includeantruntime' was not set, defaulting t
    [javac] Compiling 4 source files to C:\tools\jasperreports-5.0.1\test\classes
```

```
compilereportdesing:
      [jrc] Compiling 1 report design files.
      [jrc] log4j:WARN No appenders could be found for logger
      (net.sf.jasperreports.engine.xml.JRXmlDigesterFactory).
      [jrc] log4j:WARN Please initialize the log4j system properly.
      [jrc] log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
      [jrc] File : C:\tools\jasperreports-5.0.1\test\jasper_report_template.jrxml ... OK.

run:
     [echo] Runnin class : com.tutorialspoint.JasperReportFill
     [java] log4j:WARN No appenders could be found for logger
     (net.sf.jasperreports.extensions.ExtensionsEnvironment).
     [java] log4j:WARN Please initialize the log4j system properly.

executereport:
     [echo] Im here

BUILD SUCCESSFUL
Total time: 32 seconds
```

As a result of above execution, you will find three files sample_report.pdf, sample_report.html, sample_report.xls generated in the C:\ directory.